



## TABLE OF CONTENTS

TABLE OF CONTENTS .....	2
REAL PARTY IN INTEREST .....	3
RELATED APPEALS AND INTERFERENCES .....	3
STATUS OF THE CLAIMS .....	3
STATUS OF AMENDMENTS .....	3
SUMMARY OF THE INVENTION .....	3
ISSUE .....	5
GROUPING OF THE CLAIMS .....	5
ARGUMENT .....	5
I. <i>Hind</i> is not available to form the basis for an obviousness rejection .....	6
II. <i>Schmidt's</i> wake-up packets are not sent during a boot-up period as claimed .....	6
APPENDIX .....	9

### **REAL PARTY IN INTEREST**

The present application is assigned to International Business Machines Corporation, the real party of interest.

### **RELATED APPEALS AND INTERFERENCES**

No related appeal is presently pending.

### **STATUS OF THE CLAIMS**

Claims 1-24 stand finally rejected by the Examiner as noted in the Final Office Action dated December 4, 2003 and in the Advisory Action dated March 15, 2004.

### **STATUS OF AMENDMENTS**

No amendment was submitted subsequent to the Final Office Action dated December 4, 2003.

### **SUMMARY OF THE INVENTION**

An Universal Resource Identifier (URI) is a string of text characters for identifying an abstract or physical resource within a network environment. An URI can be further classified as a locator, a name or both. An Universal Resource Locator (URL) is a type of URI string that identifies resources via a representation of their primary access mechanism (*e.g.*, network location). URL addresses are generally served as global addresses utilized by web browsers to access documents and other resources on the Internet. As utilized herein, the Internet refers to the worldwide collection of computer networks that utilize the Transmission Control Protocol/Internet Protocol (TCP/IP) to communicate with one another. An URL specifies a protocol to be utilized for accessing a resource (such as the hypertext transport protocol for accessing a World Wide Web page), the name of a server on which the resource resides, and, optionally, the path to a particular resource (such as a hypertext markup language file) on the server. Encoded within each URL address string is the Internet Protocol address of a destination server.

The parsing of URI character strings, such as URL addresses, is often incorporated within pattern searching algorithms utilized by network processors. Such pattern search algorithms are utilized to find the longest matching binary sequence from a collection of binary strings previously stored. Specifically, a pattern searching algorithm compares an input search key with a data string that was previously stored in a database to find the longest match. The database in which data strings are stored often includes a lookup table that, after a match has been established between an input search key and a data string within the database, either retrieves information or executes a program linked to the data string.

Pattern matching searches are utilized in packet-based communication networks to facilitate routing of packets among multiple interconnected nodes. Specialized nodes called routers are responsible for delivering (or forwarding) a packet to its destination according to an IP destination address. IP currently supports a network routing protocol known as IPv4 (Internet Protocol Version 4) that specifies a 32-bit address in the header of each packet. For each packet received through an input link interface, a router reads the address field to determine the identity of the device (such as another router or a host computer) to which the packet should be forwarded before reaching its final destination. Depending on the size of the computer network and its structure, the packet is either directly forwarded to its destination or sent to another router in a similar way that a letter is passed through several post offices until reaching its final destination.

For Internet applications, a network processor determines the IP address of a destination server to which the packet is to be ultimately delivered by decoding an URL address. A network processor is typically required to handle millions of packets per second; hence, it must be capable of processing URL strings very efficiently. Conventionally, URL strings are processed incrementally with one byte at a time using a longest prefix match algorithm. The process continues to reiterate as long as more than one stored URL prefix matches the corresponding piece of the URL string from the packet being processed. Once the process has eliminated all but one of the stored URL prefixes, the single remaining prefix is utilized to identify the desired destination address.

After an optimum destination node has been determined, the router encodes the corresponding destination address into the address field of the packet and delivers the packet to a particular output link interface according to the encoded destination address. Such method of URL processing lookup has increasingly become a critical bottleneck for Internet traffic. Consequently, it would be desirable to provide an improve method for parsing and processing an URI character string such that an unique network resource can be efficiently determined.

In accordance with a preferred embodiment of the present invention, a subset of characters is initially defined as delimiters such that all remaining characters are defined as non-delimiters. Next, a search key is constructed by: (1) generating a full match search increment having the binary representation of a data string element and (2) concatenating a pattern search prefix to the full match search increment to form the search key (as shown in block 314 of Figure 3). The data string element includes a set of non-delimiters located between a pair of delimiters. The pattern search prefix is a cumulative pattern search result of all previous full match search increments. A full match search is then performed within a lookup table utilizing the search key. If a match is found in the lookup table, a new search key is constructed (as shown in block 326 of Figure 3). If no match is found in the lookup table, a previous full match search result is utilized to process the data string (as shown in block 328 of Figure 3).

### ISSUES

Is the Examiner's rejection of Claims 1-2, 4, 9-10, 12, 17-18 and 20 under 35 U.S.C. § 103(a) as being unpatentable over *Hind et al.* (US 6,463,440) in view of *Risvik* (US 6,377,945) well-founded?

### GROUPING OF THE CLAIMS

For purposes of this Appeal, Claims 1-24 stand or fall together as a single group.

### ARGUMENT

The Examiner's rejections of Claims 1-24 are not well-founded and should be reversed.

I. *Hind* is not available to form the basis for an obviousness rejection

*Hind*, on its face, is assigned to International Business Machines Corporation—the common assignee of the present application. The issue date of *Hind* is October 8, 2002, which is after the filing date of the present application (*i.e.*, July 3, 2001). Thus, according to MPEP § 706.02(l)(1), *Hind* can only be qualified as a reference under 35 U.S.C. § 102(e) but is not available to form the basis for an obviousness rejection under § 103. In addition, *Risvik* does not teach or suggest the novel features of the claimed invention.

On page 2 of the Advisory Action, the Examiner asserts that the Applicants had accepted *Hind* as a reference because the Applicants did not exclude of its usage in the response to the Examiner's first Office Action. Actually, in the response to the Examiner's first Office Action, Applicants did traverse the usage of *Hind* for the § 103 rejection by arguing on merits. Then, in the response to the Examiner's Final Office Action, Applicants traversed the usage of *Hind* for the § 103 rejection by pointing out that *Hind* can only be qualified as a reference under 35 U.S.C. § 102(e) but is not available to form the basis for an obviousness rejection under § 103 according to MPEP § 706.02(l)(1). Because the exclusion of *Hind* for the present application is statutory, Applicants can raise such issue at any time during the prosecution of the present application without being considered untimely. As such, the § 103 rejection is improper.

II. The cited references do not teach or suggest the claimed steps in Claim 1

In the alternative, Claim 1 (and similarly Claims 8 and 17) recites steps of "performing a full match search within a lookup table utilizing said search key," "in response to finding a match within said lookup table, returning to said constructing a search key," and "in response to not finding a match within said lookup table, utilizing a previous full match search result to process said data string."

On page 3 of the Final Office Action, the Examiner asserts that the above-mentioned claimed steps are disclosed by *Hind* in col. 9, line 49 - col. 10, line 40. In essence, col. 9, line 49 - col. 10, line 40 is summarized in Figure 4 of *Hind*. It is clear that Figure 4 of *Hind* does not teach or suggest the finding of a match within a lookup table and the responses to the two

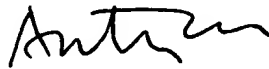
different findings. *Risvik* does not teach or suggest the claimed steps either. Because the cited references, whether considered separately or in combination, do not teach or suggest the claimed invention, the § 103 rejection is improper.

**CONCLUSION**

For the reasons stated above, Appellants believe that the claimed invention clearly is patentably distinct over the cited references and that the rejections under 35 U.S.C. § 103 are not well-founded. Hence, Appellants respectfully urge the Board to reverse the Examiner's rejection.

Please charge the IBM Deposit Account **50-0563** in the amount of \$330.00 for submission of a Brief in support of Appeal. No additional fee or extension of time is believed to be required; however, in the event an additional fee or extension of time is required, please charge that fee or extension of time requested to the IBM Deposit Account **50-0563**.

Respectfully submitted,



---

Antony P. Ng  
*Registration No. 43,427*  
DILLON & YUDELL, LLP  
8911 N. Cap. of Texas Hwy., suite 2110  
Austin, Texas 78759  
(512) 343-6116

ATTORNEY FOR APPELLANTS



## APPENDIX

1. A method for performing a pattern match search for a data string having a plurality of characters separated by delimiters, said method comprising:

defining a subset of characters as delimiters such that all remaining characters are defined as non-delimiters;

constructing a search key by:

generating a full match search increment comprising the binary representation of a data string element, wherein said data string element includes a plurality of non-delimiters between a pair of delimiters; and

concatenating a pattern search prefix to said full match search increment to form said search key, wherein said pattern search prefix is a cumulative pattern search result of all previous full match search increments;

performing a full match search within a lookup table utilizing said search key;

in response to finding a match within said lookup table, returning to said constructing a search key; and

in response to not finding a match within said lookup table, utilizing a previous full match search result to process said data string.

2. The method of claim 1, wherein said constructing a search key is preceded by pointing to a character within said data string.

3. The method of claim 2, wherein said constructing further comprises:

evaluating said character within said data string to determine whether or not said character is a delimiter;

in response to a determination that said character within said data string being a delimiter:

delivering a full match search increment into a search key register, wherein said search increment comprises a binary representation of all non-delimiters between said delimiter and an immediately preceding delimiter; and

concatenating said pattern search prefix to said search increment within said search key element;

in response to a determination that said character within said data string not being a delimiter, appending a binary representation of said character to said search increment; and

incrementing said pointer.

4. The method of claim 1, wherein said method further includes updating said pattern search prefix, in response to finding a matching patter.

5. The method of claim 1, wherein said performing a full match search further comprises:

determining whether or not a full match for said search key exists within said hash table by:

hashing said search key to produce a hash key result;

indexing a hash table utilizing said hash key result to find a matching stored pattern; and

resolving collisions in said hash table utilizing a pattern search control block.

6. The method of claim 1, wherein said data string is a Universal Resource Indicator address, and said data string element is a URI element.

7. The method of claim 6, wherein said delimiters include period characters or slash characters.

8. The method of claim 6, wherein said step of constructing a search key further includes:

scanning an IP data packet to determine a first URI element to be parsed;

initializing a URI pointer to a first character within said first URI element; and

initializing said pattern search prefix to zero.

9. A system for performing a pattern match search for a data string having a plurality of characters separated by delimiters, said system comprising:

means for defining a subset of characters as delimiters such that all remaining characters are defined as non-delimiters;

means for constructing a search key by:

generating a full match search increment comprising the binary representation of a data string element, wherein said data string element includes a plurality of non-delimiters between a pair of delimiters; and

concatenating a pattern search prefix to said full match search increment to form said search key, wherein said pattern search prefix is a cumulative pattern search result of all previous full match search increments;

means for performing a full match search within a lookup table utilizing said search key;

means for returning to said constructing a search key, in response to finding a matching pattern within said lookup table; and

means for utilizing the previous full match search result to process said data string, in response to not finding a matching pattern within said lookup table.

10. The system of claim 9, wherein said system further includes processing means for pointing to a character within said data string prior to constructing a search key.

11. The system of claim 10, wherein said processing means for constructing a search key further comprises:

means for evaluating said character to determine whether or not said character is a delimiter;

means responsive to said character being a delimiter for:

delivering a full match search increment into a search key register, wherein said search increment comprises a binary representation of all

non-delimiters between said delimiter and an immediately preceding delimiter; and

concatenating said pattern search prefix to said search increment within said search key element;

means responsive to said character not being a delimiter for appending a binary representation of said character to said search increment; and

means for incrementing said pointer.

12. The system of claim 9, wherein said system further includes means responsive to finding a matching pattern for updating said pattern search prefix.

13. The system of claim 9, wherein said means for performing a full match search further comprises:

means for determining whether or not a full match for said search key exists within said hash table by:

hashing said search key to produce a hash key result;

indexing a hash table utilizing said hash key result to find a matching stored pattern; and

resolving collisions in said hash table utilizing a pattern search control block.

14. The system of claim 9, wherein said data string is a Universal Resource Indicator address, and said data string element is a URI element.

15. The system of claim 14, wherein said delimiters include period characters or slash characters.

16. The system of claim 14, wherein said means for constructing a search key further comprises:

means for scanning an IP data packet to determine a first URI element to be parsed;

means for initializing a URI pointer to a first character within said first URI element; and

means for initializing said pattern search prefix to zero.

17. A computer program product for performing a pattern match search for a data string having a plurality of characters separated by delimiters, said computer program product comprising:

instruction means for defining a subset of characters as delimiters such that all remaining characters are defined as non-delimiters;

instruction means for constructing a search key by:

generating a full match search increment comprising the binary representation of a data string element, wherein said data string element includes a plurality of non-delimiters between a pair of delimiters; and

concatenating a pattern search prefix to said full match search increment to form said search key, wherein said pattern search prefix is a cumulative pattern search result of all previous full match search increments;

instruction means for performing a full match search within a lookup table utilizing said search key;

instruction means for returning to said constructing a search key, in response to finding a matching pattern within said lookup table; and

instruction means for utilizing the previous full match search result to process said data string, in response to not finding a matching pattern within said lookup table.

18. The computer program product of claim 17, wherein said computer program product further includes instruction means for pointing to a character within said data string prior to constructing a search key.

19. The computer program product of claim 18, wherein said instruction means for constructing a search key further includes:

instruction means for evaluating said character to determine whether or not said character is a delimiter;

instruction means responsive to said character being a delimiter for:

delivering a full match search increment into a search key register, wherein said search increment comprises a binary representation of all non-delimiters between said delimiter and an immediately preceding delimiter; and

concatenating said pattern search prefix to said search increment within said search key element;

instruction means responsive to said character not being a delimiter for appending a binary representation of said character to said search increment; and

instruction means for incrementing said pointer.

20. The computer program product of claim 17, wherein said computer program product further includes instruction means responsive to finding a matching pattern for updating said pattern search prefix.

21. (currently amended) The computer program product of claim 17, wherein said instruction means for performing a full match search further includes:

instruction means for determining whether or not a full match for said search key exists within said hash table by:

hashing said search key to produce a hash key result;

indexing a hash table utilizing said hash key result to find a matching stored pattern; and

resolving collisions in said hash table utilizing a pattern search control block.

22. The computer program product of claim 17, wherein said data string is a Universal Resource Indicator address, and said data string element is a URI element.

23. The computer program product of claim 22, wherein said delimiters include period characters or slash characters.



24. The computer program product of claim 22, wherein said instruction means for constructing a search key further includes:

instruction means for scanning an IP data packet to determine a first URI element to be parsed;

instruction means for initializing a URI pointer to a first character within said first URI element; and

instruction means for initializing said pattern search prefix to zero.